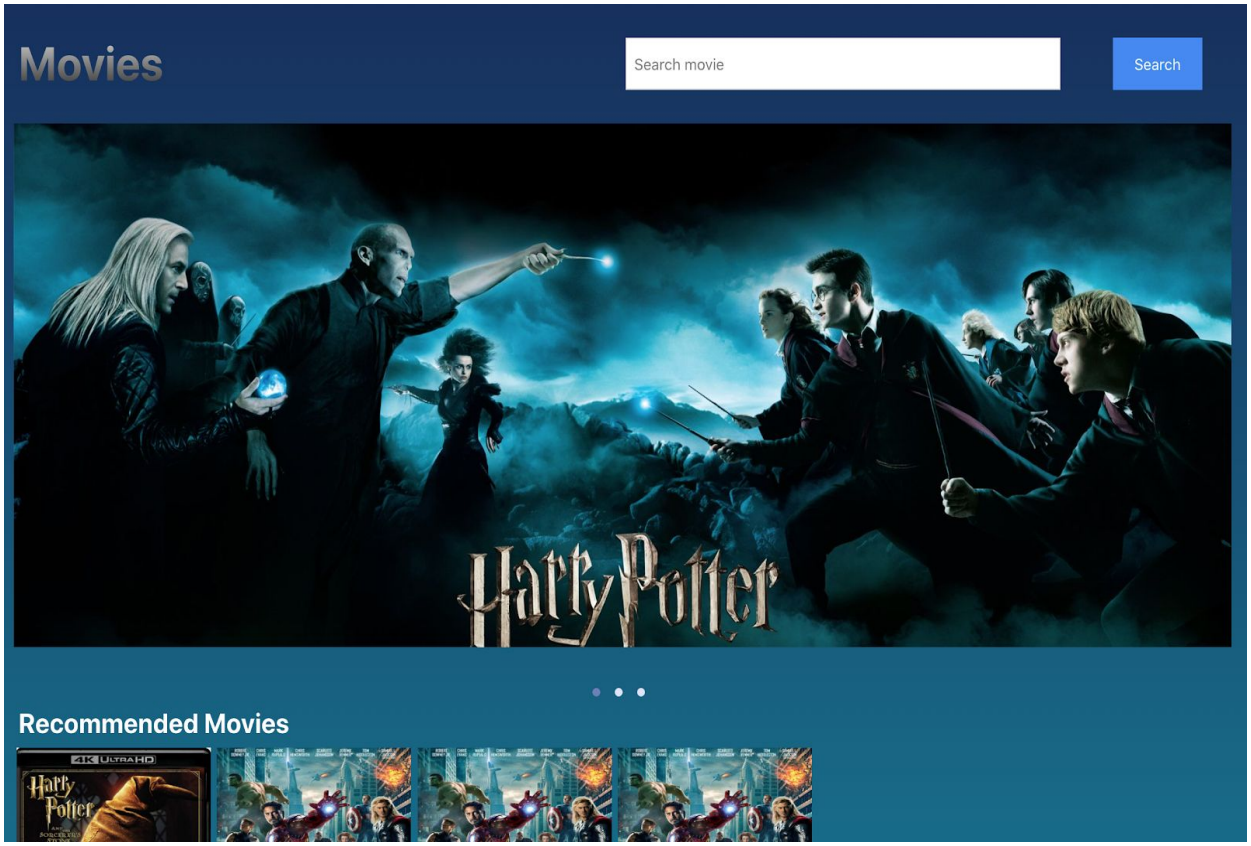


Movie Ratings

27.09.2020

Prepared By:

Vishal KALOLA
Andes Infotech
Virtual Company

Guide by:

Thomas Broussard

Introduction

1.1 Purpose

The main purpose of this document is to provide a working example of a Software Requirements Specification (SRS) in the technical aspects.

1.2 Scope

This document specifies requirements for a simple website for Movie Ratings. The application allows users to:

- Login
- Signup
- Search Movie by name
- Recommndation, Last seen and New Movie Tabs
- Show movie details
- Review and Ratings Real Time
- Some upcoming movie poster

1.3 Product perspective

1.3.1 System interfaces

The website runs in the latest version of Chrome or Firefox browser on Windows, Linux and Mac.

1.3.2 User interfaces

The Website GUI provides Search, Recommendation Movie, buttons, GridView, Scrolling, Easy to use components allowing for easy control by a keyboard and a mouse.

1.3.3 Hardware interfaces

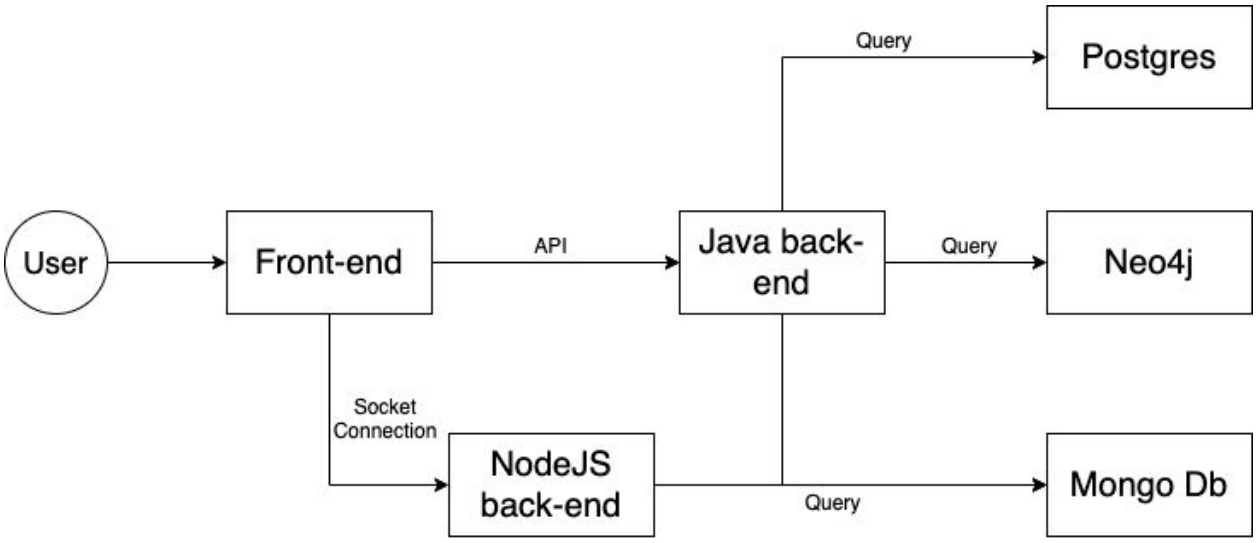
Operating system	Windows, Mac, Linux
CPU	Core 2 Quad Q6600 at 2.4 GHz or AMD Phenom 9850 at 2.5 GHz with virtualization technology
Memory	8 GB RAM
Free space	65 GB of free space

Graphics hardware	DirectX 10-compatible GPU: GeForce 9800GT 1GB or ATI Radeon HD 4870 1GB
Sound hardware	DirectX 10 compatible sound card

1.3.4 Website interfaces

Language & framework	React js, Java, Spring, Hibernate, junit, Maven, Jersey, log4j, inject, postgres, neo4j, MongoDB, Socket.io, Node js, JSON,
Tools	VSCode, Eclipse, Postgress, Neo4j, Mongodb, Docker, Docker Container, ApacheDS, APIMAN, Tomcat9.0 server, Jenkins, Sonarqube, Nexus, Virtualbox, Chrome browser

1.3.5 Website Flow



Website Flow Diagram and connection between each server

API List & Socket Connection

2.1 API

2.1.1 Login API

URL	http://localhost:8080/MovieAPI/rest/user/authenticateUser
Method	POST

Request	JSON {
	"password":"123",
	"username":"vishal1"
	}

Response	200 {
	"id": 44,
	"username": "vishal1",
	"password": "123",
	"birthdate": "2020-09-06",
	"gender": "male",
	"email": "vishalkalola196@gmail.com",
	"country": "France",
	"area": "val-de-marne",
	"city": "vitry",
	"street": "50",
	"pincode": "94800",
	"role": "user",
	"createdon": "06/20/2020 23:20:33",
	"updatedon": "06/20/2020 23:20:33"
	}

2.1.1 SignUp API

URL	http://localhost:8080/MovieAPI/rest/user/createUser
Method	POST
Request	JSON: { "password":"123", "username":"vishal3", "birthdate" : "12/05/1994",

	"gender":"male", "email":"vishalkalola196@gmail.com", "country":"france", "area":"villejuif", "city":"val-de-marne", "street":"50, avenue karl marx", "pincode":"94800", "role":"user" }

Response	200 {
	"id": 44,
	"username": "vishal1",
	"password": "123",
	"birthdate": "2020-09-06",
	"gender": "male",
	"email": "vishalkalola196@gmail.com",
	"country": "France",
	"area": "val-de-marne",
	"city": "vitry",
	"street": "50",
	"pincode": "94800",
	"role": "user",
	"createdon": "06/20/2020 23:20:33",
	"updatedon": "06/20/2020 23:20:33"
	}

2.1.1 Home API

URL	http://localhost:8080/MovieAPI/rest/MovieService/getTopMovies
Method	POST

Request	JSON: {
	"userid":"1"
	}

Response	200 {
	"recommondationmovie": [
	{

```
            "id": 0,
            "title": "Harry potter Philosopher's Stone",
            "details": "Harry Potter and the Philosopher's Stone is a fantasy novel written by British author J. K. Rowling. The first novel in the Harry Potter series and Rowling's debut novel.",
            "imageLink": "https://www.linkpicture.com/q/harrypotter1.jpg",
            "releaseDate": "26 June 1997",
            "category": "Science Friction",
            "movieDirector": "J. K. Rowling",
            "createdon": "26/02/2020 01:02:29",
            "updatedon": "26/02/2020 01:02:29"
        }
    ],
    "lastSeenMovies": [
        {
            "id": 0,
            "title": "Harry potter Philosopher's Stone",
            "details": "Harry Potter and the Philosopher's Stone is a fantasy novel written by British author J. K. Rowling. ",
            "imageLink": "https://www.linkpicture.com/q/harrypotter1.jpg",
            "releaseDate": "26 June 1997",
            "category": "Science Friction",
            "movieDirector": "J. K. Rowling",
            "createdon": "26/02/2020 01:02:29",
            "updatedon": "26/02/2020 01:02:29"
        },
    ],
    "newMovie": [
        {
            "id": 6,
            "title": "Avenger age of ultron",
            "details": "This is good movie",
            "imageLink": "https://www.linkpicture.com/q/avenger.png",
            "releaseDate": "12/05/1994",
            "category": "Science Friction",
```

	"movieDirector": "vishal KALOLA", "createdon": "20/29/2020 12:29:16", "updatedon": "20/29/2020 12:29:16" }] }

2.1.1 Search API

URL	http://localhost:8080/MovieAPI/rest/MovieServicesearchMovie?name=Av
Method	GET

Response	200 [
	{
	"id": 4,
	"title": "Avenger",
	"details": "This is good movie",
	"imageLink": "https://www.linkpicture.com/q/avenger.png",
	"releaseDate": "12/05/1994",
	"category": "Science Friction",
	"movieDirector": "vishal KALOLA",
	"createdon": "20/27/2020 12:27:44",
	"updatedon": "20/27/2020 12:27:44"
	},
	{
	"id": 5,
	"title": "Avenger end game",
	"details": "This is good movie",
	"imageLink": "https://www.linkpicture.com/q/avenger.png",
	"releaseDate": "12/05/1994",
	"category": "Science Friction",
	"movieDirector": "vishal KALOLA",
	"createdon": "20/28/2020 12:28:46",
	"updatedon": "20/28/2020 12:28:46"
	}
	]

2.1.1 Details API

URL	http://localhost:8080/MovieAPI/rest/MovieService/detailsMovie?id=4
Method	GET

Response	200 {
	"id": 4,
	"title": "Avenger",
	"details": "This is good movie",
	"imageLink": "https://www.linkpicture.com/q/avenger.png",
	"releaseDate": "12/05/1994",
	"category": "Science Friction",
	"movieDirector": "vishal KALOLA",
	"createdon": "20/27/2020 12:27:44",
	"updatedon": "20/27/2020 12:27:44"
	}

2.2 Socket Connection:

2.2.1 addcomment

Name	addcomment

Request	JOSN: {
	"title": “vishal1”,
	"Comment": “This is good movie”,
	"userid":”1”,
	"movieid": “4”
	}

Response	200: {"status":200,"msg":"getComment 1"}

2.2.1 SeenMovie

Name	SeenMovie

Request	JSON: {
	"userid": “1”,
	"movieid": “4”
	}

Response	200: {"status":200,"msg":"Added Records"} 500: {"status":500,"err":"already gave ratings"}

2.2.1 addratings

Name	addratings

Request	JSON: {
	"rating":`5`,
	"userid": '1',
	"movieid": '4'
	}

Response	200: {"status":200,"msg":"Reached Rating"} 500: {"status":500,"err":"already gave ratings"} 500: {"status":500,"err":"Send body data also"}

2.2.1 getComment

Name	getComment
Request	getComment1 subscribe channel with movie id

Response	**Get History :**
	
	200: [{
	"title":"vishal1",
	"Comment":"this is nice movie",
	"Userid":"1",
	"Movieid":"4",
	"createdon":"10/10/2020 10:20:20",
	"updatedon":"10/10/2020 10:20:20"
	}, {
	"title":"vishal1",
	"Comment":"this is nice movie",
	"Userid":"1",
	"Movieid":"4",
	"createdon":"10/10/2020 10:20:20",
	"updatedon":"10/10/2020 10:20:20"
	}]
	
	Single Comment :
	
	200: {
	"title":"vishal1",
	"Comment":"this is nice movie",
	"Userid":"1",
	"Movieid":"4",

	“createdon”:”10/10/2020 10:20:20”, “updatedon”:”10/10/2020 10:20:20” }

2.2.1 getRatings

Name	getRatings
Request	getRatings1 subscribe channel with movie id

Response	200: {
	"status":200,
	"Ratings":4.5,
	"Isenable":true,
	"Totalcount":1000,
	"Five":900,
	"Four":20,
	"Three":20,
	"Two":20,
	"One":40
	}

2.2.1 connection

Name	connection
Request	String: { query:`movieId=”4”&userid=”1”`}
Response	Subscribe “Connect” method for get successful connection

2.2.1 disconnect

Name	disconnect
Request	String: { query:`movieId=”4”&userid=”1”`}
Response	Subscribe “disconnect” method for get successful connection

Database Structure

3.1 Postgres

3.1.1 User Table

	id [PK] bigint	createdon character varying (255)	password character varying (255)	role character varying (255)	updatedon character varying (255)	username character varying (255)
1	41	06/17/2020 23:17:27	MTIz	user	06/17/2020 23:17:27	vishal
2	44	06/20/2020 23:20:33	MTIz	user	06/20/2020 23:20:33	vishal1
3	47	06/21/2020 23:21:25	MTIz	user	06/21/2020 23:21:25	vishal2

3.1.2 Address Table

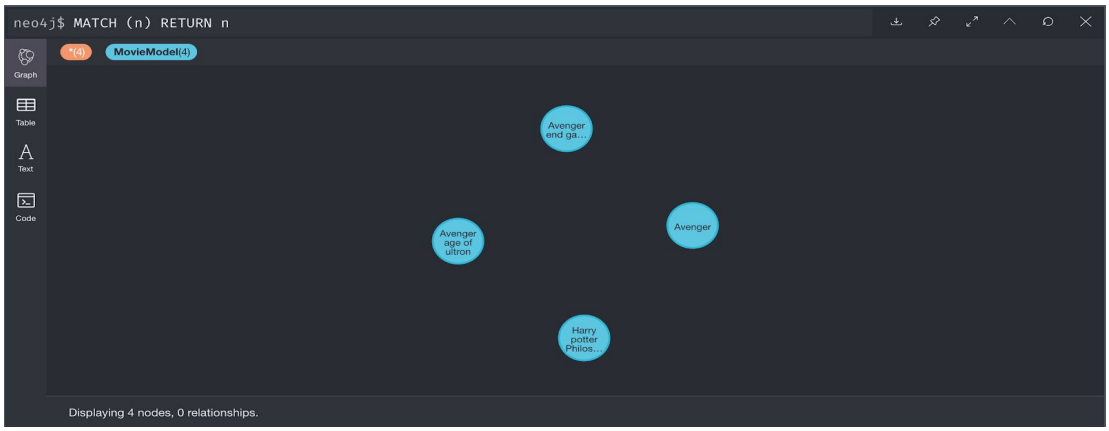
	id [PK] bigint	birthdate character varying (255)	createdon character varying (255)	email character varying (255)	gender character varying (255)	updatedon character varying (255)	fk_user bigint
1	42	2020-09-06	06/17/2020 23:17:27	vishalkalola196@gmail.com	male	06/17/2020 23:17:27	41
2	45	2020-09-06	06/20/2020 23:20:33	vishalkalola196@gmail.com	male	06/20/2020 23:20:33	44
3	48	2020-08-09	06/21/2020 23:21:25	vishalkalola196@gmail.com	male	06/21/2020 23:21:25	47

3.1.3 Contact Table

	id [PK] bigint	area character varying (255)	city character varying (255)	country character varying (255)	createdon character varying (255)	pincode character varying (255)	street character varying (255)	updatedon character varying (255)	fk_user bigint
1	43	val-de-marne	vitry	France	06/17/2020 23:17:27	94800	50	06/17/2020 23:17:27	41
2	46	val-de-marne	vitry	France	06/20/2020 23:20:33	94800	50	06/20/2020 23:20:33	44
3	49	val-de-marne	vitry	France	06/21/2020 23:21:25	94800	50	06/21/2020 23:21:25	47

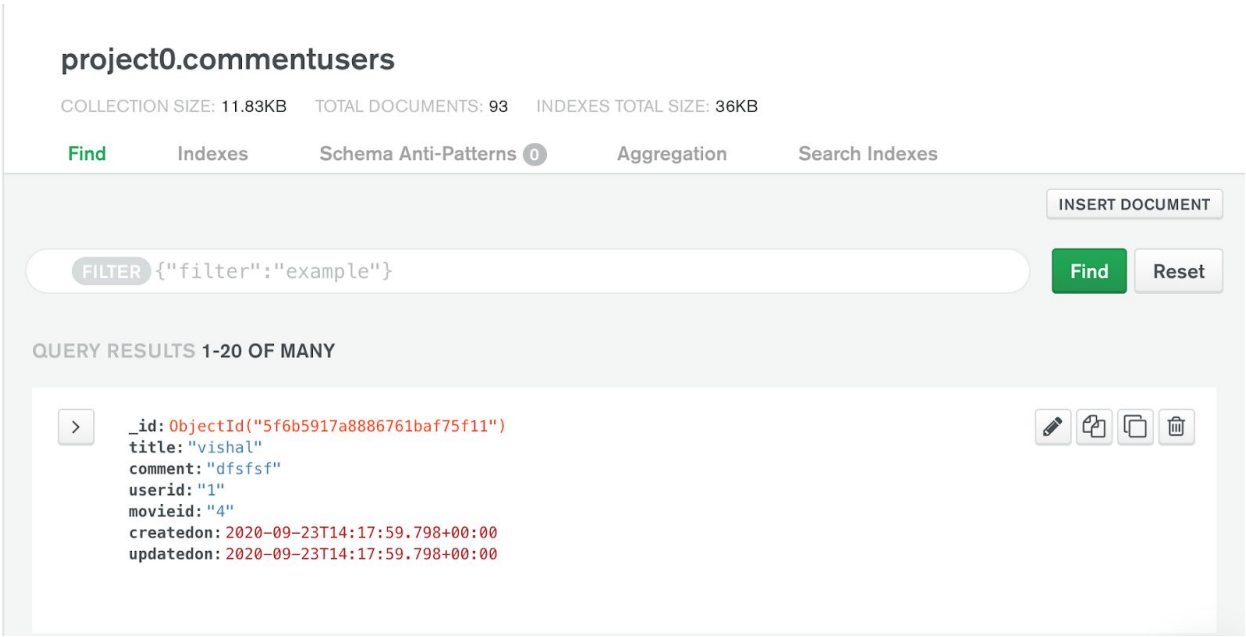
3.2 Neo4j

3.2.1 MovieModel

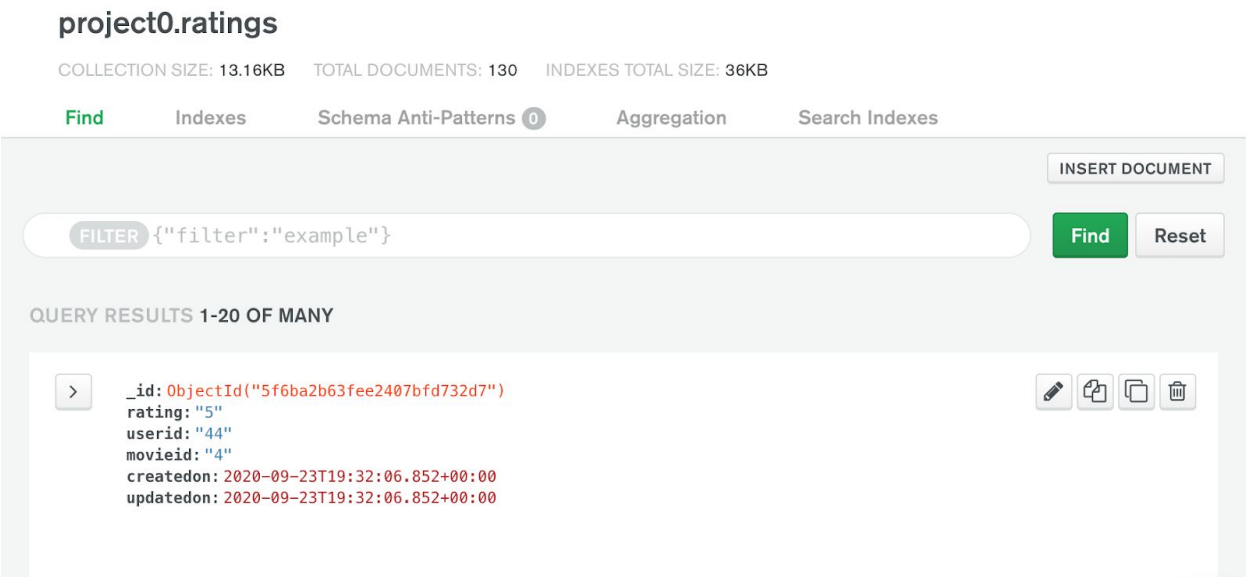


3.3 Postgress

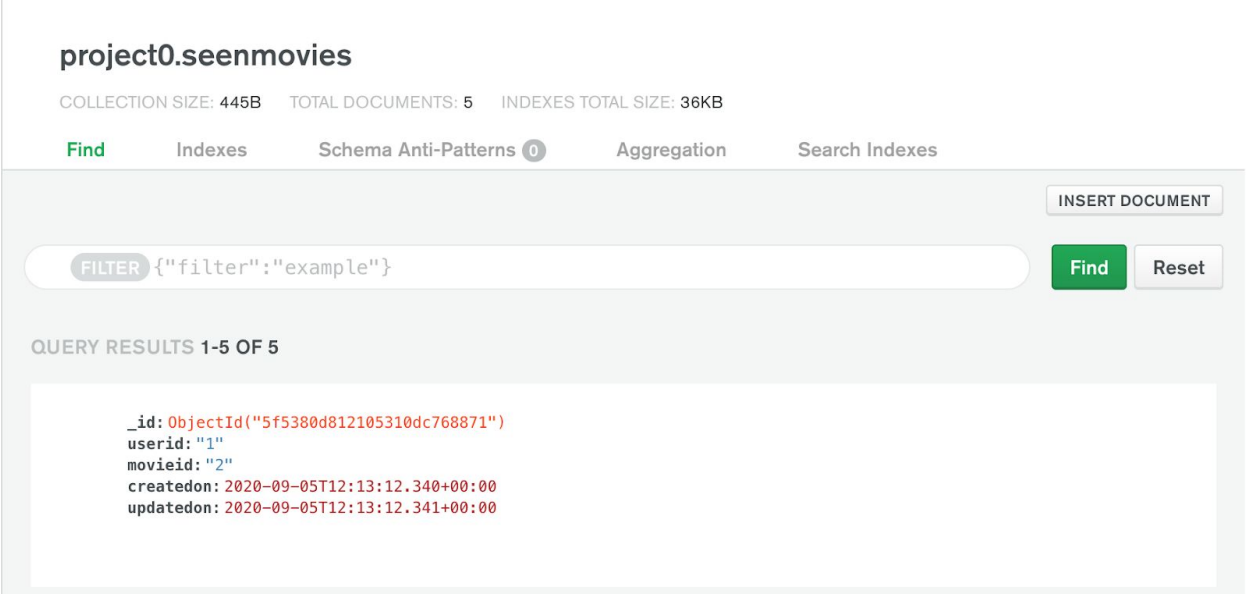
3.3.1 CommentUsers



3.3.2 ratings



3.3.3 SeenMovies



Version Control

5.1 Github

5.1.1 Movie Rating Frontend

https://github.com/vishalkalola1/MovieFrontend.git

5.1.2 Movie backend Java

https://github.com/vishalkalola1/MovieBackend.git

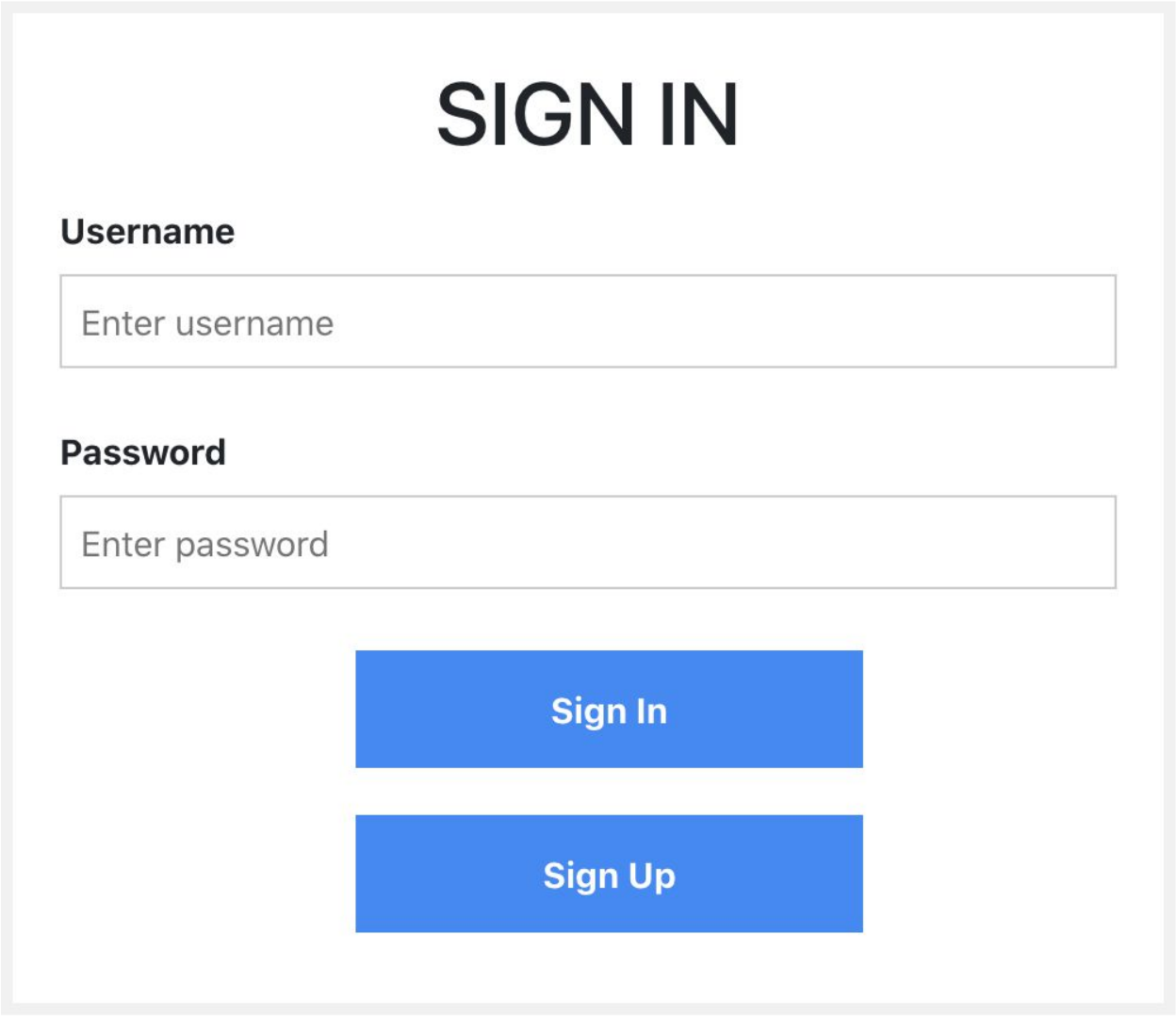
5.1.3 Movie backend Node js

https://github.com/vishalkalola1/MovieBackendNode.git

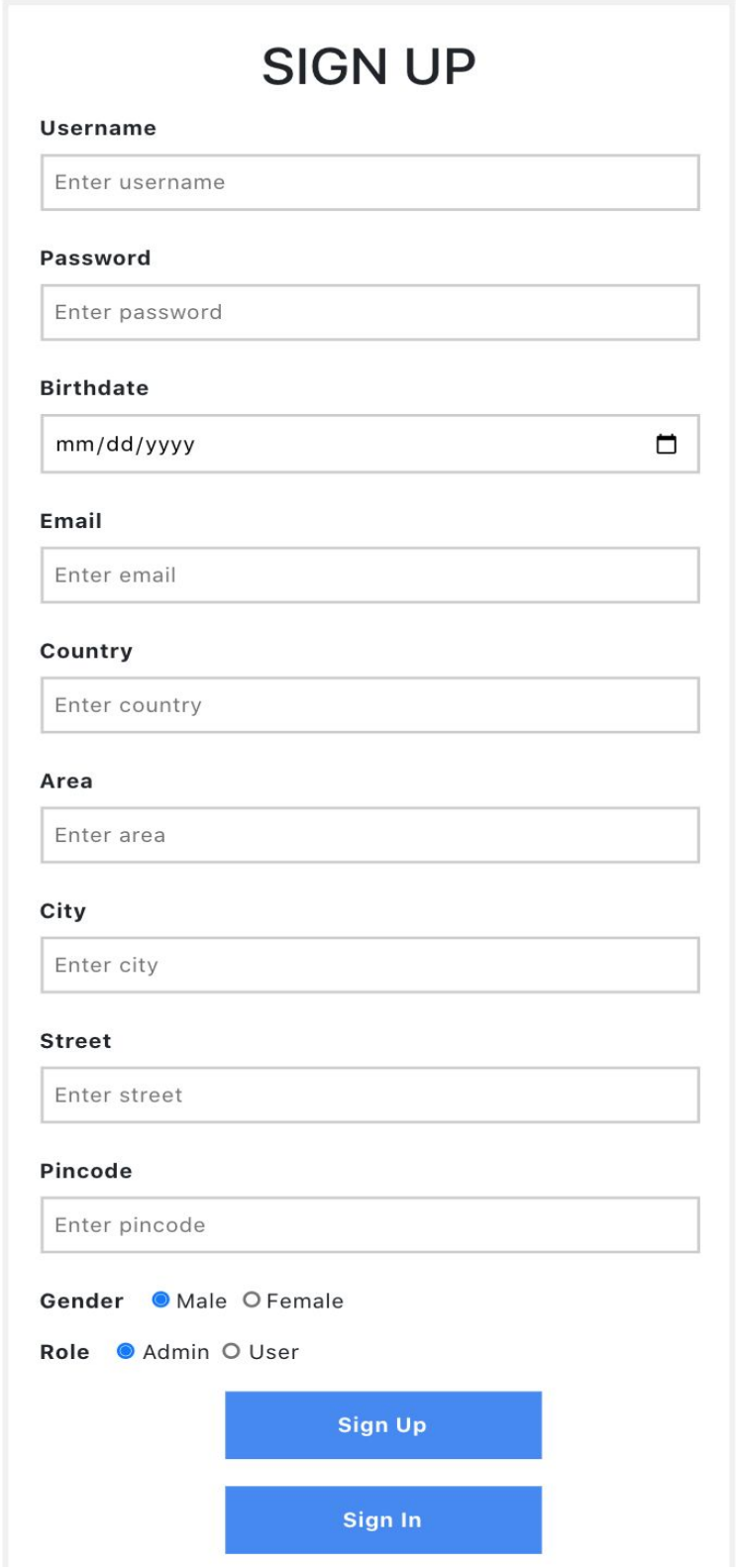
Screenshots & Video

6.1 Website

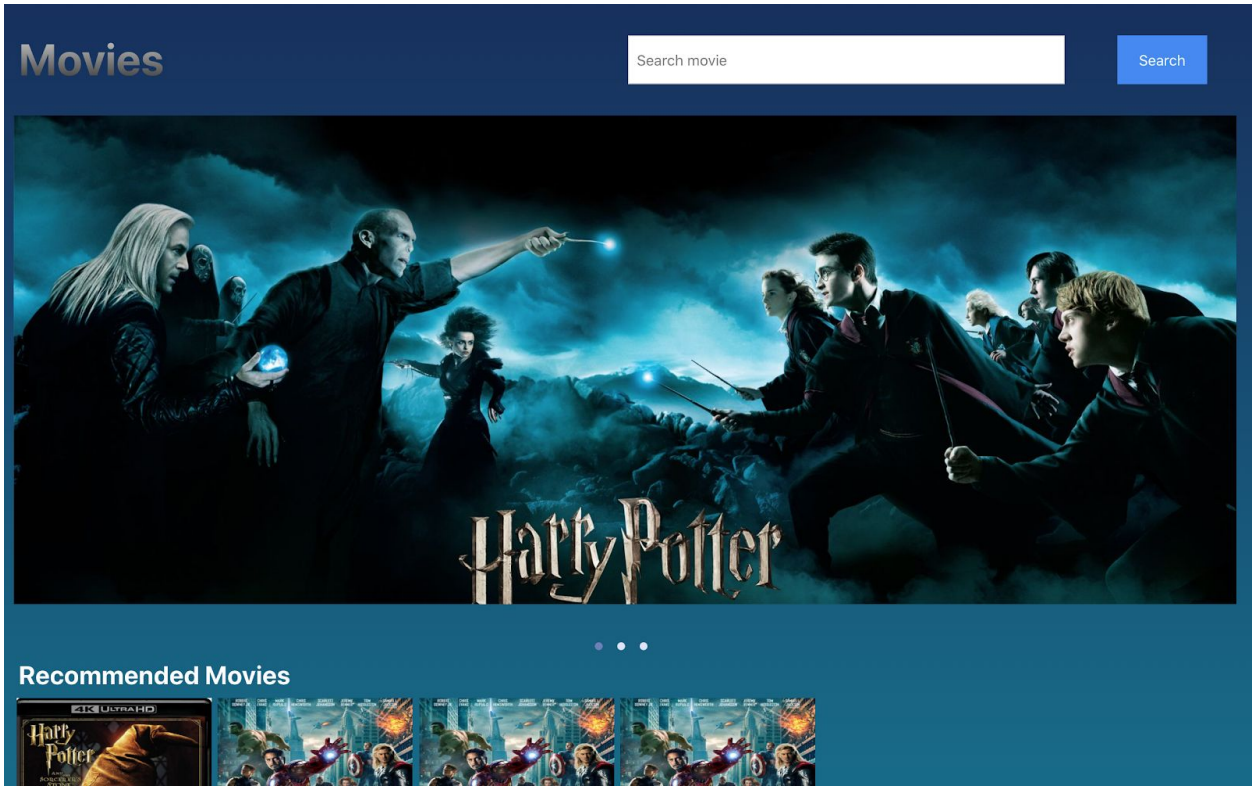
6.1.1 Login

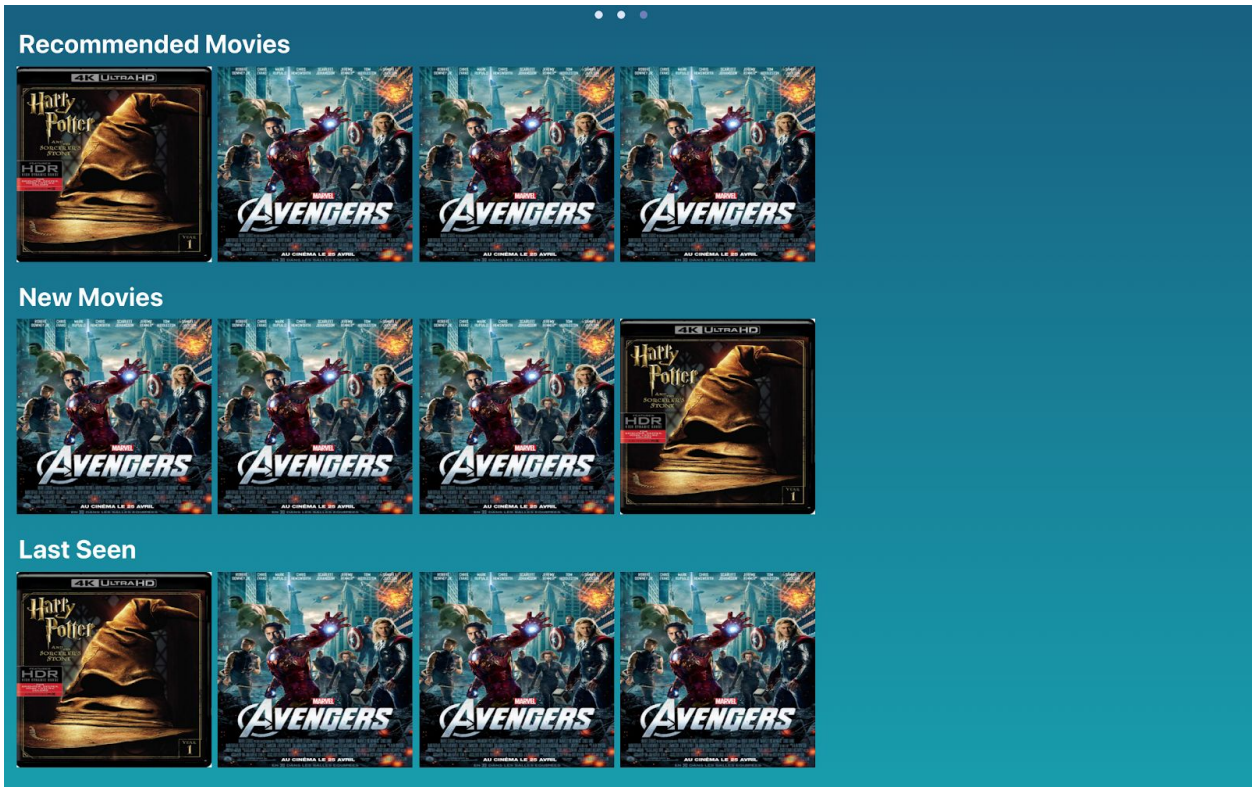


6.1.2 Sign Up

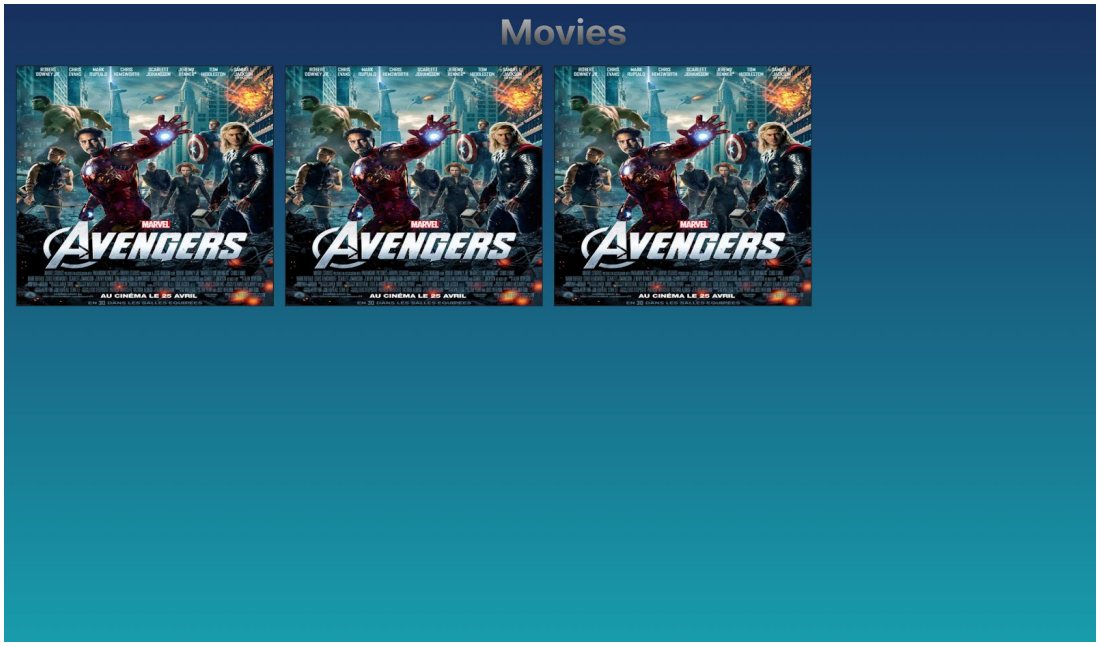

SIGN UP
Username
Enter username
Password
Enter password
Birthdate
mm/dd/yyyy
Email
Enter email
Country
Enter country
Area
Enter area
City
Enter city
Street
Enter street
Pincode
Enter pincode
Gender
Male
Female
Role
Admin
User
Sign Up
Sign In

6.1.3 Home Page

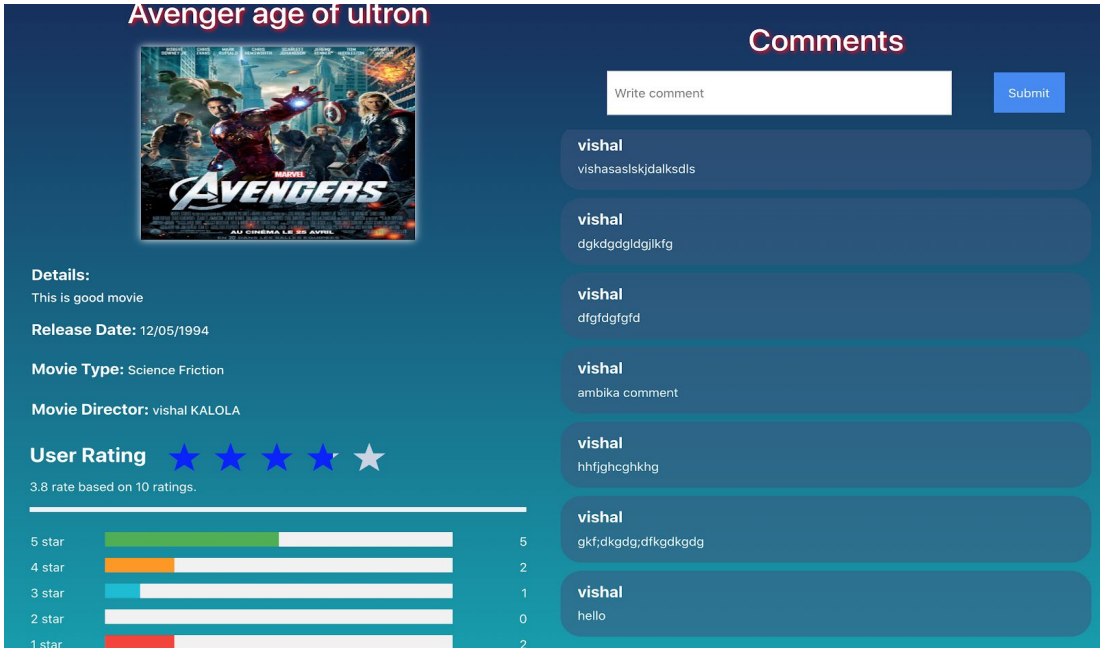




6.1.4 Search Movie



6.1.5 Movie details



6.1.6 Video Link

https://drive.google.com/file/d/10xvZQBtTSSFKtXo3koR9hzgSOymUED-F/view?usp=sharing

Thank You